University Politehnica of Bucharest
Faculty of Automatic Control and Computers
Computer Science and Engineering Department

Diploma Thesis

Mobile Gateway for Wireless Sensor Networks utilizing drones

by

Ioan Deaconu

Supervisor: Prof. Dr. Ing. Nicolae Ţăpuş
Supervisor: As. Drd. Ing. Andrei Voinescu

Bucharest, September 2014

Contents

Abstract

Keywords Wireless Sensor Networks, task, scheduling, graph cuts

Acknowledgements

Chapter 1

Introduction

Thesis Intro - no more than 3 pages.

Chapter 2

Related Work

Related work for task scheduling

Chapter 3

Hardware Platform

In this chapter we will present the hardware platforms uses in order
chesti ..... help andrei

3.1 The Parrot AR.Drone 2.0

Parrot AR.Drone is a wifi radio controlled flying quadcopter built by the French company Parrot. The original drone was released in 2010 and in 2012 it was replaced by version 2.0. Since the launch of the original AR.Drone, more the half a milion units have been sold, making it one of the, if not, the most popular drone on the market.

The reason of its success is not entirely due to the relatively low price of around 300$ but also because it is very easy to learn how to control the drone and also because of the usb port that accomodate any device using that interface and the linux operating system

Figure 3.1: *The arrot AR.Drone 2.0*

Because of those reasons, the Drone has a number of aftermaket modules that can be attached to it like the Flight Recorder GPS Module. This module has a built in storage of 4GB for video recording purposes and a built in GPS receiver. This allows the drone to follow a predetermined path of waypoints and to return back from where it took off automatically, all within the limit of the Wi-Fi connection with the control device.

The arrot AR.Drone 2.0 specifications are :

- 1GHz 32 bit ARM Cortex A8 processor with 800MHz video DSP TMS320DMC64x
- Linux 2.6.32
- 1Gbit DDR2 RAM at 200MHz
- USB 2.0 high speed for extensions

- Wi-Fi b,g,n
- 3 axis gyroscope 2000/second precision
- 3 axis accelerometer +-50mg precision
- 3 axis magnetometer 6 precision
- Pressure sensor +/- 10 Pa precision
- Ultrasound sensors for ground altitude measurement
- 60 fps vertical QVGA camera for ground speed measurement
- 30 fps 720p front mounted camera

3.2 The Sparrow Dongle

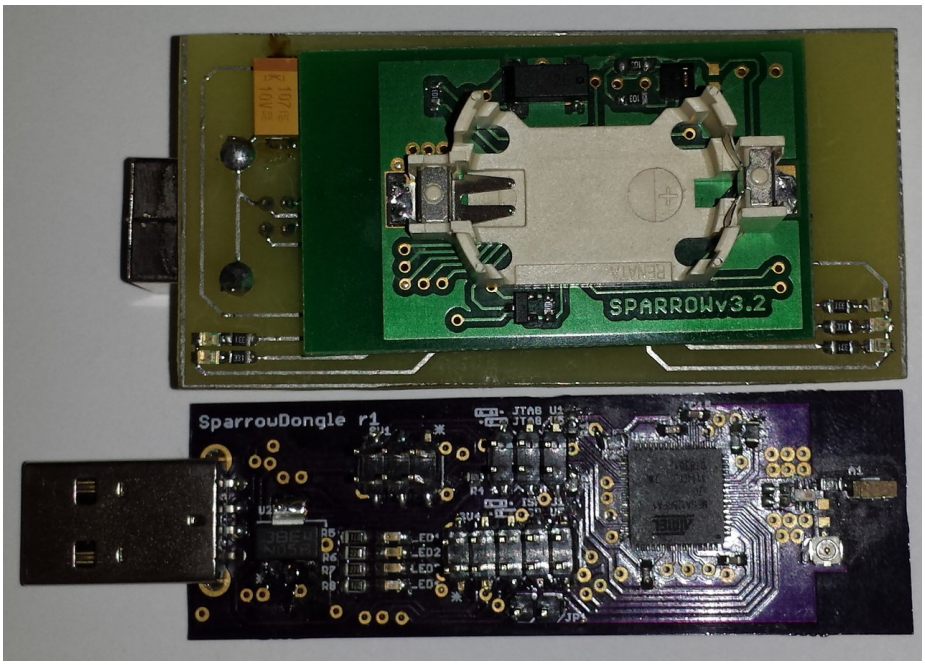

Figure 3.2: *The Sparrow Dongle next to the SparrowV32*

3.3 The SparrowV32

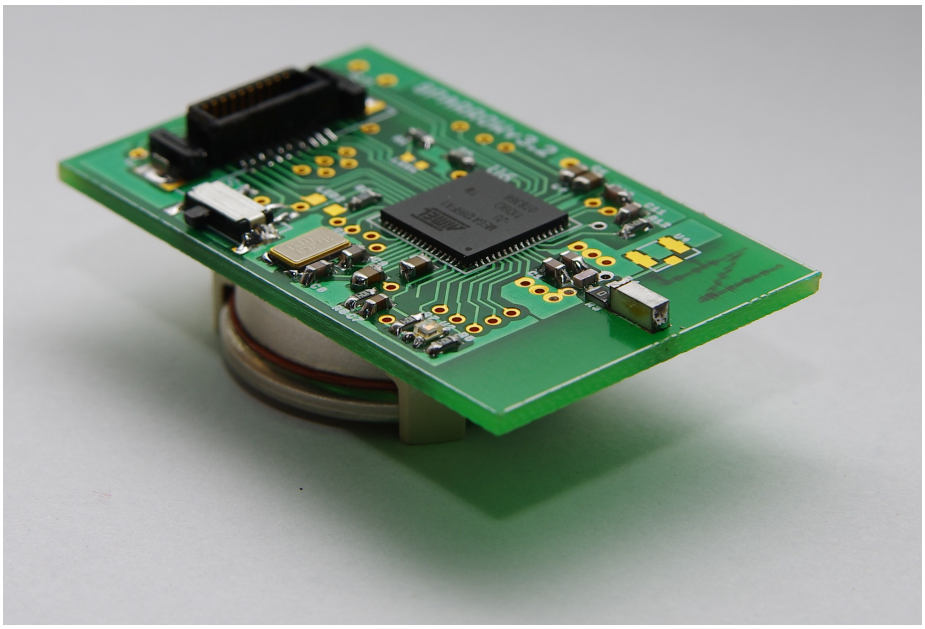

Figure 3.3: *The SparrowV32*

Chapter 4

Software Environment: Contiki Operating System

software intro ....

The software is composed on diferent interlocking (trebuie modificat) modules running on diferent devices and operating systems.

The modules are written mainly in java and c.

again ... andrei ?

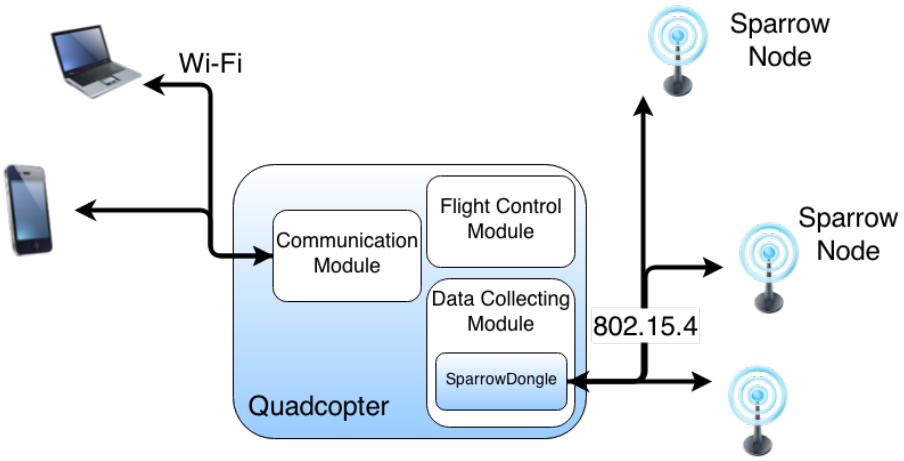

Figure 4.1: *Modules and connections between them and devices*

4.1 The Data Collecting Module

The module saves the collected data int-o the drones internal memory and prases the data in order to obtain certain informations like number of nodes currently connected to the dongle, de signal strength etc. This informations are passed to the communication module to provide to the user realtime feedback.

The memory area in which the informations sent to the user are saved is shared between this module and the comunication module. Basically, the way this two modules interract with each other can be compared to the consumer - producer problem, where the Data Collecting Module can be associated with the producer side and the Communication Module with the consumer side.

The main problem consists of deadlocks and data starvation. This is prevent with the use of one mutex that allows only one thread at a time to moddify the informations.

Listing 4.1: Data Collection use of mutex

```
pthread_mutex_lock(&data_lock);
add_node_data(get_current_timestamp(),read_data + 7);
pthread_mutex_unlock(&data_lock);
```

The mutex is used similarly in the Communication Module when it consumes the information.

Listing 4.2: Data Collection use of mutex

```
void add_node_data(long long time_stamp , char *p) {
    int i;
    int id = get_hex(p,2);

    /* the power of the signal calculated in dB */
    int power = -90 + 3* (get_hex(p + 64,2)-1);


    /* creating a file with unique name */
    char file_name[100];
    sprintf(file_name, "/node_logs/%lli_%i",file_timestamp,id);
    /* opening a file in append mode */
    FILE *fptr = fopen(file_name,"a");

    /* saving the new data at the end of the file */
    fprintf(fptr,"%s",p);
    fclose(fptr);

    /* searching for previous connection of the same node*/
```

```
    for(i = 0 ;i < node_nr;i++) {
        if(data[i].id == id) {
            /* timestamp update - node is stil reachable and sending data *
            data[i].time_stamp = time_stamp;
            return;
        }
    }

    /* new node found by the drone */
    data[node_nr].id = id;
    data[node_nr].time_stamp = time_stamp;
    node_nr ++;
}
```

4.2 The Communication Module

4.3 The Saved Data Transfer Module

Bibliography

[Atm] Atmel. Rzraven hardware user's guide. [cited at p. -]

[Cal04] Edgar H. Callaway. *Wireless Sensor Networks: Architectures and Protocols.* CRC Press, 2004. [cited at p. -]

[CK03] C. Chong and S.P. Kumar. In *Sensor networks: Evolution, opportunities, and challenges*, 2003. [cited at p. -]

[DPdR+05] Flvia Coimbra Delicato, Fbio Protti, Jos Ferreira de Rezende, Luiz F. Rust da Costa Carmo, and Luci Pirmez. Application-driven node management in multihop wireless sensor networks. In Pascal Lorenz and Petre Dini, editors, *ICN (1)*, volume 3420 of *Lecture Notes in Computer Science*, pages 569–576. Springer, 2005. [cited at p. -]

[ea07] Roberto Verdone et al. *Wireless Sensor and Actuator Networks: Technologies, Analysis and Design.* Academic Press, 1st edition, 2007. [cited at p. -]

[GH88] Olivier Goldschmidt and Dorit S. Hochbaum. Polynomial algorithm for the k-cut problem. 1988. [cited at p. -]

[HJ05] Tarek Hagras and Jan Janecek. A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems. *Parallel Computing*, 31(7):653–670, 2005. [cited at p. -]

[Jac06] Brian Jacokes. Lecture notes on multiway cuts and k-cuts, July 2006. [cited at p. -]

[KD06] Snehal Kamalapur and Neeta Deshpande. Efficient cpu scheduling: A genetic algorithm based approach. *Ad Hoc and Ubiquitous Computing*, pages 206 – 207, December 2006. [cited at p. -]

[Li08] Xiang-Yang Li. *Wireless Ad Hoc and Sensors Networks: Theory and Applications.* Cambridge University Press, 2008. [cited at p. -]

[NGT99] Andrew Goldberg Nec, Andrew V. Goldberg, and Kostas Tsioutsiouliklis. Cut tree algorithms. In *In Symposium on Discrete Algorithms*, pages 376–385, 1999. [cited at p. -]

[PB05] C. Prehofer and C. Bettstetter. Self-organization in communication networks: principles and design paradigms. *IEEE Communications Magazine*, 43(7):78–85, July 2005. [cited at p. -]

[TEÖ07] Yuan Tian, Eylem Ekici, and Füsun Özgüner. Energy-constrained task mapping and scheduling in wireless sensor networks. 2007. [cited at p. -]

[YJK07] William M Iversen Yunseop (James) Kim, Robert G Evans. The future of intelligent agriculture. *Resource*, October 2007. [cited at p. -]

[zig] Zigbit wireless modules datasheet. [cited at p. -]

Appendix A

Contiki API

A.1 Process macros

- `PROCESS_THREAD (name, ev, data )` - Define the body of a process. This macro is used to define the body (protothread) of a process. The process is called whenever an event occurs in the system, A process always starts with the PROCESS_BEGIN() macro and end with the PROCESS_END() macro.
- `PROCESS_BEGIN ()` - Define the beginning of a process.
- `PROCESS_END ()` - Define the end of a process.
- `PROCESS_YIELD ()` - Yields the currently running process
- `PROCESS_WAIT_EVENT_UNTIL (c)` - Wait for an event to be posted to the process, with an extra condition. This macro is very similar to PROCESS_WAIT_EVENT() in that it blocks the currently running process until the process receives an event. But PROCESS_WAIT_EVENT_UNTIL() takes an extra condition which must be true for the process to continue.
- `PROCESS_PAUSE` - Yield the process for a short while. This macro yields the currently running process for a short while, thus letting other processes run before the process continues.

A.2 uIP functions

- `PSOCK_INIT (psock, buffer, buffersize)` - Initializes a proto-socket. This macro initializes a protosocket and must be called before the pro-

tosocket is used. The initialization also specifies the input buffer for the protosocket.

- `PSOCK_SEND (psock, data, datalen)` - Send data. This macro sends data over a protosocket. The protosocket protothread blocks until all data has been sent and is known to have been received by the remote end of the TCP connection.
- `PSOCK_READBUF (psock)` - Read data until the buffer is full. This macro will block waiting for data and read the data into the input buffer specified with the call to PSOCK_INIT(). Data is read until the buffer is full..
- `CCIF process_event_t tcpip_event` - The uIP event. This event is posted to a process whenever a uIP event has occurred.
- `CCIF void tcp_listen (u16_t port)` - Open a TCP port. This function opens a TCP port for listening. When a TCP connection request occurs for the port, the process will be sent a tcpip_event with the new connection request.
- `struct uip_conn *tcp_connect(uipipaddr_t *ripaddr,u16 port, void *appstate)` - This function opens a TCP connection to the specified port at the host specified with an IP address. Additionally, an opaque pointer can be attached to the connection. This pointer will be sent together with uIP events to the process.
- `uip_connected()` - Has the connection just been connected?
- `uip_closed()` - Has the connection been closed by the other end?
- `uip_aborted()` - Has the connection been aborted by the other end?
- `uip_timedout()` - Has the connection timed out?
- `uip_newdata()` - Is new incoming data available?
- `uip_close()` - Close the current connection.

Appendix B

Node capabilities

Task	AVR Raven™	Sparrow	Sparrow Power
Temperature sensing	✔	✔	
Humidity sensing		✔	
Voltage & Current sensing			✔
Event detection	✔	✔	✔
Alarm beep	✔		
LED signal	✔	✔	

Table B.1: Node capabilities

List of Figures

List of Tables

Listings